REMARKS

This is a full and timely response to the outstanding non-final Office Action mailed November 24, 2008 (Paper No. 20081007). Upon entry of this response, claims 1-23 are pending in the application. In this response, claims 1, 8, 13, and 23 are amended. Applicant respectfully requests entry of the amendments herein and reconsideration of all pending claims.

I. Rejection of Claims 1-12 under 35 U.S.C. § 112, Second Paragraph

Claims 1-12 have been rejected under 35 U.S.C. §112, second paragraph, as allegedly being indefinite for failing to particularly point out and distinctly claim the subject matter which the Applicant regards as his invention. Specifically, in connection with independent claims 1 and 8 the Office Action alleges (p. 2) that "it is unclear how to registering is triggered to detect the physical address space is released, when the device has been released" (emphasis in original). Independent claims 1 and 8 are amended to recite "the indication occurs responsive to the physical address space being released". Applicant submits that the amendment overcomes the rejection of claims 1-12, and requests that the rejection be withdrawn.

II. Rejection of Claim 23 under 35 U.S.C. § 101

Claim 23 stands rejected under 35 U.S.C. §101 as allegedly being directed to non-statutory subject matter. Specifically, the Office Action (p. 1) alleges that "claim 23 is not limited to tangible embodiments". Claim 23 is amended to recite "computer readable storage medium", and Applicant requests that the rejection be withdrawn.

III. Rejection of Claims 1-23 under 35 U.S.C. §103

Claims 1-23 are rejected under §103(a) as allegedly obvious over *Armilli* (U.S. 6,907,494) in view of *Browning et al.* (U.S. 6,918,023). Applicant respectfully traverses this rejection. It is well established at law that, for a proper rejection of a claim under 35 U.S.C. §103 as being obvious based upon a combination of references, the cited combination of references

must disclose, teach, or suggest (either implicitly or explicitly) all elements/features/steps of the claim at issue. *See, e.g., In re Dow Chemical*, 5 U.S.P.Q.2d 1529, 1531 (Fed. Cir. 1988); *In re Keller*, 208 U.S.P.Q.2d 871, 881 (C.C.P.A. 1981).

A. Independent Claim 1

1. The proposed combination does not teach "provide an indication in a virtual memory data structure associated with the process"

The Office Action acknowledges (pp. 3-4) that *Armilli* does not specifically teach this feature, but contends that *Browning et al.* teaches this feature as follows:

> However, Browning teaches a method for invalidating specified pre-translations maintained in a data processing system which maintains decentralized copies of pre-translations, wherein the data processing system comprising a DMA mapping agent to map a virtual buffer to physical address (steps 802-8 16, figure 8), when a detection on memory removal operation is being process (col. 7 lines 21 through col. 8 line 44), and also defined in figure 8 that determining whether the RPN entry for the virtual memory page is valid (step 824), when flag is set on memory removal process (step 834), i.e., registering is triggered by detection that the physical address space that was being used by process associated with the device is being released, and the virtual memory page is maintained for mapping as defined in step 828, and further comprising the steps of to register **by providing an indication in the virtual memory data structure for the process that the virtual address space, previously available to the process, is no longer valid for process use, as defined in steps 824-830 in figure 8, and (step 910, figure 9 and col. 8 lines 58-64, i.e., scan all registered RPN lists and invalidates all entries, including virtual address space, corresponding to real pages that within the range of memory to be removed)**, and (col. 8 lines 53-58, i.e., receiving acknowledgement of interrupt from all CPUs and then triggering to register by detecting that real pages that are within range of memory to be removed).
> (Office Action, p. 4, emphasis added.)

Applicant respectfully disagrees with this characterization of *Browning et al.*, and submits that none of the passages in *Browning et al.* that are relied on in the rejection disclose a "virtual memory data structure associated with ***the process***".

It appears that the Office Action relies on the RPN (real page number) data structure as teaching the claimed "virtual memory data structure", and Applicant assumes (for the sake of argument) that the RPN is a virtual data memory structure. Even so, *Browning et al.* does not

teach or suggest that the RPN is "associated with the process" as recited in claim 1. In fact, *Browning et al.* appears to suggest that the RPN is not associated with a particular process:

> The RPN lists are private copies of virtual to physical translations. The lists are maintained in memory descriptors that are associated and maintained with virtual buffers. Pointers to virtual buffers may be passed from one entity to another. When a pointer to a virtual buffer is passed from one entity to another, control of the RPN list included in the buffer's descriptor is thus passed from one entity to another. In this manner, the lists are not owned by any particular entity. An entity may be a software process, such as a routine, application, or operating system function, or it may be a subsystem.
> (*Browning et al.*, Col. 4, lines 3-13.)

If the rejection is maintained in another Office Action, the Examiner is requested to point out with particularity which data structure in *Browning et al.* allegedly corresponds to a "virtual memory data structure associated with the process".

2. The proposed combination does not teach "provide an indication…that the virtual address space, previously available to the process, is no longer valid for use by the process"

The Office Action acknowledges (pp. 3-4) that *Armilli* does not specifically teach this feature, but contends that *Browning et al.* teaches this feature as follows:

> However, Browning teaches a method for invalidating specified pre-translations maintained in a data processing system which maintains decentralized copies of pre-translations, wherein the data processing system comprising a DMA mapping agent to map a virtual buffer to physical address (steps 802-8 16, figure 8), when a detection on memory removal operation is being process (col. 7 lines 21 through col. 8 line 44), and also defined in figure 8 that determining whether the RPN entry for the virtual memory page is valid (step 824), when flag is set on memory removal process (step 834), i.e., registering is triggered by detection that the physical address space that was being used by process associated with the device is being released, and the virtual memory page is maintained for mapping as defined in step 828, and further comprising the steps of to register **by providing an indication in the virtual memory data structure for the process that the virtual address space, previously available to the process, is no longer valid for process use, as defined in steps 824-830 in figure 8, and (step 910, figure 9 and col. 8 lines 58-64, i.e., scan all registered RPN lists and invalidates all entries, including virtual address space, corresponding to real pages that within the range of memory to be removed)**, and (col. 8 lines 53-58, i.e., receiving acknowledgement of interrupt from all CPUs and then triggering to register by detecting that real pages that are within range of memory to be removed).

(Office Action, p. 4, emphasis added.)

Applicant respectfully disagrees with this characterization of *Browning et al.* It appears that the Office Action relies on the RPN (real page number) data structure as teaching the claimed "virtual memory data structure", and Applicant assumes (for the sake of argument) that the RPN is a virtual data memory structure. Even so, *Browning et al.* does not teach or suggest that the RPN provides an indication that "that the virtual address space, previously available to the process, is no longer valid" as recited in claim 1.

Browning et al. appears to disclose that an RPN is associated with virtual addresses, through the memory descriptor 50 which includes the RPN (See FIG. 3A.) *Browning et al.* also describes scanning entries in multiple RPN lists and invalidating these entries as part of a kernel memory remove operation. (Col. 8, lines 45-50.) However, Applicant submits that an invalid RPN entry is not an indication that the virtual address space associated with the RPN is no longer valid. Instead, an invalid RPN entry merely signifies that the virtual-to-physical mapping contained in the RPN is invalid such that the mapping must be updated. (See FIG. 8, blocks 824 and 832: if RPN entry is invalid, retranslate buffer page.)

> The RPN lists are private copies of virtual to physical translations. The lists are maintained in memory descriptors that are associated and maintained with virtual buffers. Pointers to virtual buffers may be passed from one entity to another. When a pointer to a virtual buffer is passed from one entity to another, control of the RPN list included in the buffer's descriptor is thus passed from one entity to another. In this manner, the lists are not owned by any particular entity. An entity may be a software process, such as a routine, application, or operating system function, or it may be a subsystem.
> (*Browning et al.*, Col. 4, lines 1-13.)

If the rejection is maintained in another Office Action, the Examiner is requested to clearly explain how the RPN in *Browning et al.* allegedly indicates "that the virtual address space, previously available to the process, is no longer valid".

3. The proposed combination does not teach "wherein registering the indication is triggered by detection that the physical address space…has been released"

The Office Action acknowledges (pp. 3-4) that *Armilli* does not specifically teach this feature, but contends that *Browning et al.* teaches this feature as follows:

> However, Browning teaches a method for invalidating specified pre-translations maintained in a data processing system which maintains decentralized copies of pre-translations, wherein the data processing system comprising a DMA mapping agent to map a virtual buffer to physical address (steps 802-816, figure 8), ***when a detection on memory removal operation is being process (col. 7 lines 21 through col. 8 line 44), and also defined in figure 8 that determining whether the RPN entry for the virtual memory page is valid (step 824), when flag is set on memory removal process (step 834), i.e., registering is triggered by detection that the physical address space that was being used by process associated with the device is being released,*** and the virtual memory page is maintained for mapping as defined in step 828, and further comprising the steps of to register by providing an indication in the virtual memory data structure for the process that the virtual address space, previously available to the process, is no longer valid for process use, as defined in steps 824-830 in figure 8, and (step 910, figure 9 and col. 8 lines 58-64, i.e., scan all registered RPN lists and invalidates all entries, including virtual address space, corresponding to real pages that within the range of memory to be removed), and (col. 8 lines 53-58, i.e., receiving acknowledgement of interrupt from all CPUs and then triggering to register by detecting that real pages that are within range of memory to be removed).
> (Office Action, p. 4, emphasis added.)

Applicant respectfully disagrees with this characterization of *Browning et al.* as teaching "wherein registering the indication is triggered by detection that the physical address space…has been released". The Office Action specifically refers to steps 824 and 834 of FIG. 8 for this teaching. With regard to step 824, Applicant respectfully submits that checking the validity of an RPN entry has nothing to do with detecting the release of physical addresses. With regard to step 834, Applicant respectfully submits that the "is memory remove in progress flag set" flag checked in step 834 is not the same as detecting that the physical address space has been released. Instead, this flag merely indicates that process 900 (FIG. 9) is executing. Furthermore, the action that occurs if the flag is set is initializion of DMA mapping (step 828). Even assuming (for the sake of argument) that initialization of DMA mapping could be properly

understood as an indication, initialization of DMA mapping is certainly not the specific indication recited in claim 1, namely "providing an indication in a virtual memory data structure associated with the process that the virtual address space, previously available to the process, is no longer valid for use by the process".

4. Conclusion

As explained above, the proposed combination of *Armilli* in view of *Browning et al.* does not teach at least the features noted above and recited in claim 1. Therefore, a *prima facie* case establishing an obviousness rejection has not been made, and the rejection should be withdrawn.

B. Independent Claim 8

1. The proposed combination does not teach "providing an indication in a virtual memory data structure for the process"

The Office Action acknowledges (p. 6) that *Armilli* does not specifically teach this feature, but contends that *Browning et al.* teaches this feature as follows:

> However, Browning teaches a method for invalidating specified pre-translations maintained in a data processing system which maintains decentralized copies of pre-translations, wherein the data processing system comprising a DMA mapping agent to map a virtual buffer to physical address (steps 802-816, figure 8), when a detection on memory removal operation is being process (col. 7 lines 21 through col. 8 line 44), and also defined in figure 8 that determining whether the RPN entry for the virtual memory page is valid (step 824), when flag is set on memory removal process (step 834), i.e., registering is triggered by detection that the physical address space that was being used by process associated with the device is being released, and the virtual memory page is maintained for mapping as defined in step 828, and further comprising the steps of to register **by providing an indication in the virtual memory data structure for the process that the virtual address space, previously available to the process, is no longer valid for process use, as defined in steps 824-830 in figure 8, and (step 910, figure 9 and col. 8 lines 58-64, i.e., scan all registered RPN lists and invalidates all entries, including virtual address space, corresponding to real pages that within the range of memory to be removed)**, and (col. 8 lines 53-58, i.e., receiving acknowledgement of interrupt from all CPUs and then triggering to register by detecting that real pages that are within range of memory to be removed).
> (Office Action, pp. 6-7, emphasis added.)

Applicant respectfully disagrees with this characterization of *Browning et al.*, and submits that none of the passages in *Browning et al.* that are relied on in the rejection disclose a "virtual memory data structure for ***the process***".

It appears that the Office Action relies on the RPN (real page number) data structure as teaching the claimed "virtual memory data structure", and Applicant assumes (for the sake of argument) that the RPN is a virtual data memory structure. Even so, *Browning et al.* does not teach or suggest that the RPN is "for the process" as recited in claim 8. In fact, *Browning et al.* appears to suggest that the RPN is <u>not</u> for a process:

> The RPN lists are private copies of virtual to physical translations. The lists are maintained in memory descriptors that are associated and maintained with virtual buffers. Pointers to virtual buffers may be passed from one entity to another. When a pointer to a virtual buffer is passed from one entity to another, control of the RPN list included in the buffer's descriptor is thus passed from one entity to another. In this manner, the lists are not owned by any particular entity. An entity may be a software process, such as a routine, application, or operating system function, or it may be a subsystem.
> (*Browning et al.*, Col. 4, lines 1-13.)

If the rejection is maintained in another Office Action, the Examiner is requested to point out with particularity which data structure in *Browning et al.* allegedly corresponds to a "virtual memory data structure for the process".

2. <u>The proposed combination does not teach "providing an indication…that the virtual address space is no longer available to the process"</u>

The Office Action acknowledges (p. 6) that *Armilli* does not specifically teach this feature, but contends that *Browning et al.* teaches this feature as follows:

> However, Browning teaches a method for invalidating specified pre-translations maintained in a data processing system which maintains decentralized copies of pre-translations, wherein the data processing system comprising a DMA mapping agent to map a virtual buffer to physical address (steps 802-8 16, figure 8), when a detection on memory removal operation is being process (col. 7 lines 21 through col. 8 line 44), and also defined in figure 8 that determining whether the RPN entry for the virtual memory page is valid (step 824), when flag is set on memory removal process (step 834), i.e., registering is triggered by detection that the physical address space that was being used by process associated with the device is being released, and the virtual memory page is

> maintained for mapping as defined in step 828, and further comprising the steps of to register **by providing an indication in the virtual memory data structure for the process that the virtual address space, previously available to the process, is no longer valid for process use, as defined in steps 824-830 in figure 8, and (step 910, figure 9 and col. 8 lines 58-64, i.e., scan all registered RPN lists and invalidates all entries, including virtual address space, corresponding to real pages that within the range of memory to be removed)**, and (col. 8 lines 53-58, i.e., receiving acknowledgement of interrupt from all CPUs and then triggering to register by detecting that real pages that are within range of memory to be removed).
> (Office Action, pp. 6-7, emphasis added.)

Applicant respectfully disagrees with this characterization of *Browning et al.* It appears that the Office Action relies on the RPN (real page number) data structure as teaching the claimed "virtual memory data structure", and Applicant assumes (for the sake of argument) that the RPN is a virtual data memory structure. Even so, *Browning et al.* does not teach or suggest that the RPN provides an indication that "that the virtual address space is no longer available to the process" as recited in claim 8.

Browning et al. appears to disclose that an RPN is associated with virtual addresses, through the memory descriptor 50 which includes the RPN (See FIG. 3A.) *Browning et al.* also describes scanning entries in multiple RPN lists and invalidating these entries as part of a kernel memory remove operation. (Col. 8, lines 45-50.) However, Applicant submits that an invalid RPN entry is not an indication that the virtual address space associated with the RPN is no longer available to a process. Instead, an invalid RPN entry merely signifies that the virtual-to-physical mapping contained in the RPN is invalid such that the mapping must be updated. (See FIG. 8, blocks 824 and 832: if RPN entry is invalid, retranslate buffer page.)

> The RPN lists are private copies of virtual to physical translations. The lists are maintained in memory descriptors that are associated and maintained with virtual buffers. Pointers to virtual buffers may be passed from one entity to another. When a pointer to a virtual buffer is passed from one entity to another, control of the RPN list included in the buffer's descriptor is thus passed from one entity to another. In this manner, the lists are not owned by any particular entity. An entity may be a software process, such as a routine, application, or operating system function, or it may be a subsystem.
> (*Browning et al.*, Col. 4, lines 1-13.)

If the rejection is maintained in another Office Action, the Examiner is requested to clearly explain how the RPN in *Browning et al.* allegedly indicates "that the virtual address space is no longer available to the process".

3. The proposed combination does not teach "wherein to register is triggered by detection that the physical address space…has been released"

The Office Action acknowledges (p. 6) that *Armilli* does not specifically teach this feature, but contends that *Browning et al.* teaches this feature as follows:

> However, Browning teaches a method for invalidating specified pre-translations maintained in a data processing system which maintains decentralized copies of pre-translations, wherein the data processing system comprising a DMA mapping agent to map a virtual buffer to physical address (steps 802-816, figure 8), ***when a detection on memory removal operation is being process (col. 7 lines 21 through col. 8 line 44), and also defined in figure 8 that determining whether the RPN entry for the virtual memory page is valid (step 824), when flag is set on memory removal process (step 834), i.e., registering is triggered by detection that the physical address space that was being used by process associated with the device is being released,*** and the virtual memory page is maintained for mapping as defined in step 828, and further comprising the steps of to register by providing an indication in the virtual memory data structure for the process that the virtual address space, previously available to the process, is no longer valid for process use, as defined in steps 824-830 in figure 8, and (step 910, figure 9 and col. 8 lines 58-64, i.e., scan all registered RPN lists and invalidates all entries, including virtual address space, corresponding to real pages that within the range of memory to be removed), and (col. 8 lines 53-58, i.e., receiving acknowledgement of interrupt from all CPUs and then triggering to register by detecting that real pages that are within range of memory to be removed).
> (Office Action, pp. 6-7, emphasis added.)

Applicant respectfully disagrees with this characterization of *Browning et al.* as teaching "wherein to register is triggered by detection that the physical address space that was being used by processes associated with the device has been released". The Office Action specifically refers to steps 824 and 834 of FIG. 8 for this teaching. With regard to step 824, Applicant respectfully submits that checking the validity of an RPN entry has nothing to do with detecting the release of physical addresses. With regard to step 834, Applicant respectfully submits that the "is memory remove in progress flag set" in step 834 is not the same as detecting that the

physical address space has been released. Instead, this flag merely indicates that process 900 (FIG. 9) is executing. Furthermore, the action that occurs if the flag is set is initialization of DMA mapping (step 828). Even assuming (for the sake of argument) that initialization of DMA mapping could be properly understood as an indication, initialization of DMA mapping is certainly not the specific indication recited in claim 8, namely "providing an indication in a virtual memory data structure for the process that the virtual address space is no longer available to the process".

4. Conclusion

As explained above, the proposed combination of *Armilli* in view of *Browning et al.* does not teach at least the features noted above and recited in claim 8. Therefore, a *prima facie* case establishing an obviousness rejection has not been made, and the rejection should be withdrawn.

C. Independent Claim 13

Applicant respectfully submits that claim 13 is allowable for at least the reason that the proposed combination of *Armilli* in view of *Browning et al.* does not disclose, teach, or suggest "means for unmapping a virtual address space for the process". The Office Action contends (pp. 8-9) that *Armilli* teaches this feature as follows: "abstract and col. 7 line 32 through col. 9 line 10, i.e., the processor's move engine works in conjunction with the associated mapping engine to take the associated memory module offline, read as un-mapping a virtual address space". Applicant disagrees with this contention. As clearly explained in the portion of *Armilli* relied on by the rejection, the mapping engine works with real addresses, not with virtual addresses:

> As seen in FIG. 3, mapping engine 26 contains a register 301 having a field 302 containing the current real address of memory module M1, and a field 304 containing the new real address for memory module M1. Similarly, mapping engine 36 contains a register 305 having field 306 containing the current real address of memory module M2 and a field 308 containing the new real address of memory module M2. Mapping engine 46 contains a register having field 310 containing the current real address

> of memory module M3 and a field 312 containing the new real address of memory module M3.
>
> Move engine 28 loads each register 301, 305, 309 as necessary to perform the memory re-configuration. Field 302 shows that memory module M1's current real address is RA1. Field 304 contains the new real address for memory module M1, and shows that it remains the same at RA1. Mapping engine 36 contains field 306, showing the current real address of memory module M1 as RA2. For its new real address, memory module M2 is given a real address that is outside the total real address space currently allocated to the physical memory system. Thus, for example, field 308 contains a new real address for memory module M2, that is RA4, which is outside the current real address space (e.g. 0-128 GB) as mapped to the real addresses (i.e. RA1-RA2). Similarly, move engine 28 has assigned memory module M3 the previous real address of memory module M2, as shown in field 312, indicating memory module M3's new real address as RA2. Field 310 shows memory module M3's current real address of RA3, which is now outside the addressable real space for the operating system.
> (Col. 7, line 65 to Col. 8 line 26.)

Furthermore, even assuming (for the sake of argument) that move engine 28 and/or mapping engine 26 does unmap a virtual address space, the unmapping has nothing to do with, and does not take into consideration, a specific process. In contrast, the embodiment as defined by claim 13 includes a "means for unmapping a virtual address space for the process". Finally, *Browning et al.* does not cure this deficiency. Even assuming (for the sake of argument) that *Browning et al.* unmaps a virtual address space, the unmapping is not performed for ***a process*** (as explained above in section III.A.1). Since the proposed combination of *Armilli* in view of *Browning et al.* does not teach all the features recited in claim 19, a *prima facie* case establishing an obviousness rejection has not been made, and the rejection should be withdrawn.

D. Independent Claim 19

In rejecting independent claim 19, the Office Action (p. 11) states that "the limitations of the claim are rejected as the same reasons as set forth in claim 8" even though a quick perusal of the claims shows they are not coextensive in scope. For example, claim 19 recites "mapping a representation of an object associated with the process in a virtual memory data structure

associated with the process", and this feature is not present in claim 8. Since the Office Action fails to address all the features recited in claim 19, the rejection of the claims 19-21 is incomplete and deficient. This rejection should be withdrawn and a new rejection issued in another ***non-final Office Action.***

Nonetheless, in the interest of advancing prosecution, Applicant now discusses some distinctions between claim 19 and the proposed combination of *Armilli* in view of *Browning et al.* In doing so, Applicant makes reference to the rejection of claim 8.

1. The proposed combination does not teach "providing an indication in the virtual memory data structure for the process"

The Office Action acknowledges (p. 6) that *Armilli* does not specifically teach this feature, but contends that *Browning et al.* teaches this feature as follows:

> However, Browning teaches a method for invalidating specified pre-translations maintained in a data processing system which maintains decentralized copies of pre-translations, wherein the data processing system comprising a DMA mapping agent to map a virtual buffer to physical address (steps 802-816, figure 8), when a detection on memory removal operation is being process (col. 7 lines 21 through col. 8 line 44), and also defined in figure 8 that determining whether the RPN entry for the virtual memory page is valid (step 824), when flag is set on memory removal process (step 834), i.e., registering is triggered by detection that the physical address space that was being used by process associated with the device is being released, and the virtual memory page is maintained for mapping as defined in step 828, and further comprising the steps of to register **by providing an indication in the virtual memory data structure for the process that the virtual address space, previously available to the process, is no longer valid for process use, as defined in steps 824-830 in figure 8, and (step 910, figure 9 and col. 8 lines 58-64, i.e., scan all registered RPN lists and invalidates all entries, including virtual address space, corresponding to real pages that within the range of memory to be removed)**, and (col. 8 lines 53-58, i.e., receiving acknowledgement of interrupt from all CPUs and then triggering to register by detecting that real pages that are within range of memory to be removed).
> (Office Action, pp. 6-7, emphasis added.)

Applicant respectfully disagrees with this characterization of *Browning et al.*, and submits that none of the passages in *Browning et al.* that are relied on in the rejection disclose a "virtual memory data structure for ***the process***".

It appears that the Office Action relies on the RPN (real page number) data structure as teaching the claimed "virtual memory data structure", and Applicant assumes (for the sake of argument) that the RPN is a virtual data memory structure. Even so, *Browning et al.* does not teach or suggest that the RPN is "for the process" as recited in claim 19. In fact, *Browning et al.* appears to suggest that the RPN is not for a particular process:

> The RPN lists are private copies of virtual to physical translations. The lists are maintained in memory descriptors that are associated and maintained with virtual buffers. Pointers to virtual buffers may be passed from one entity to another. When a pointer to a virtual buffer is passed from one entity to another, control of the RPN list included in the buffer's descriptor is thus passed from one entity to another. In this manner, the lists are not owned by any particular entity. An entity may be a software process, such as a routine, application, or operating system function, or it may be a subsystem.
> (*Browning et al.*, Col. 4, lines 3-13.)

If the rejection is maintained in another Office Action, the Examiner is requested to point out with particularity which data structure in *Browning et al.* allegedly corresponds to "virtual memory data structure for the process".

2. The proposed combination does not teach "providing an indication…that a virtual address space is no longer available for use by the process"

The Office Action acknowledges (p. 6) that *Armilli* does not specifically teach this feature, but contends that *Browning et al.* teaches this feature as follows:

> However, Browning teaches a method for invalidating specified pre-translations maintained in a data processing system which maintains decentralized copies of pre-translations, wherein the data processing system comprising a DMA mapping agent to map a virtual buffer to physical address (steps 802-8 16, figure 8), when a detection on memory removal operation is being process (col. 7 lines 21 through col. 8 line 44), and also defined in figure 8 that determining whether the RPN entry for the virtual memory page is valid (step 824), when flag is set on memory removal process (step 834), i.e., registering is triggered by detection that the physical address space that was being used by process associated with the device is being released, and the virtual memory page is maintained for mapping as defined in step 828, and further comprising the steps of to register **by providing an indication in the virtual memory data structure for the process that the virtual address space, previously available to the process, is no longer valid for process use, as defined in steps 824-830 in figure 8, and (step 910, figure 9 and col. 8 lines 58-64, i.e., scan all registered RPN lists and**

> **invalidates all entries, including virtual address space, corresponding to real pages that within the range of memory to be removed)**, and (col. 8 lines 53-58, i.e., receiving acknowledgement of interrupt from all CPUs and then triggering to register by detecting that real pages that are within range of memory to be removed).
> (Office Action, pp. 6-7, emphasis added.)

Applicant respectfully disagrees with this characterization of *Browning et al.* It appears that the Office Action relies on the RPN (real page number) data structure as teaching the claimed "virtual memory data structure", and Applicant assumes (for the sake of argument) that the RPN is a virtual data memory structure. Even so, *Browning et al.* does not teach or suggest that the RPN provides an indication that "that a virtual address space is no longer available for use by the process" as recited in claim 19.

Browning et al. appears to disclose that an RPN is associated with virtual addresses, through the memory descriptor 50 which includes the RPN (See FIG. 3A.) *Browning et al.* also describes scanning entries in multiple RPN lists and invalidating these entries as part of a kernel memory remove operation. (Col. 8, lines 45-50.) However, Applicant submits that an invalid RPN entry is not an indication that the virtual address space associated with the RPN is no longer available for use. Instead, an invalid RPN entry merely signifies that the virtual-to-physical mapping contained in the RPN is invalid such that the mapping must be updated. (See FIG. 8, blocks 824 and 832: if RPN entry is invalid, retranslate buffer page.)

> The RPN lists are private copies of virtual to physical translations. The lists are maintained in memory descriptors that are associated and maintained with virtual buffers. Pointers to virtual buffers may be passed from one entity to another. When a pointer to a virtual buffer is passed from one entity to another, control of the RPN list included in the buffer's descriptor is thus passed from one entity to another. In this manner, the lists are not owned by any particular entity. An entity may be a software process, such as a routine, application, or operating system function, or it may be a subsystem.
> (*Browning et al.*, Col. 4, lines 3-13.)

If the rejection is maintained in another Office Action, the Examiner is requested to clearly explain how the RPN in *Browning et al.* allegedly indicates "that a virtual address space is no longer available for use by the process".

3. The proposed combination does not teach "as triggered by detection of a physical address space…being released"

The Office Action acknowledges (pp. 3-4) that *Armilli* does not specifically teach this feature, but contends that *Browning et al.* teaches this feature as follows:

> However, Browning teaches a method for invalidating specified pre-translations maintained in a data processing system which maintains decentralized copies of pre-translations, wherein the data processing system comprising a DMA mapping agent to map a virtual buffer to physical address (steps 802-816, figure 8), ***when a detection on memory removal operation is being process (col. 7 lines 21 through col. 8 line 44), and also defined in figure 8 that determining whether the RPN entry for the virtual memory page is valid (step 824), when flag is set on memory removal process (step 834), i.e., registering is triggered by detection that the physical address space that was being used by process associated with the device is being released,*** and the virtual memory page is maintained for mapping as defined in step 828, and further comprising the steps of to register by providing an indication in the virtual memory data structure for the process that the virtual address space, previously available to the process, is no longer valid for process use, as defined in steps 824-830 in figure 8, and (step 910, figure 9 and col. 8 lines 58-64, i.e., scan all registered RPN lists and invalidates all entries, including virtual address space, corresponding to real pages that within the range of memory to be removed), and (col. 8 lines 53-58, i.e., receiving acknowledgement of interrupt from all CPUs and then triggering to register by detecting that real pages that are within range of memory to be removed).
> (Office Action, p. 4, emphasis added.)

Applicant respectfully disagrees with this characterization of *Browning et al.* as teaching "as triggered by detection of a physical address space used by the process being released". The Office Action specifically refers to steps 824 and 834 of FIG. 8 for this teaching. With regard to step 824, Applicant respectfully submits that checking the validity of an RPN entry has nothing to do with detecting the release of physical addresses. With regard to step 834, Applicant respectfully submits that the "is memory remove in progress flag set" flag checked in step 834 is not the same as detecting that the physical address space is being released. Instead, this flag merely indicates that process 900 (FIG. 9) is executing. Furthermore, the action that occurs if the flag is set is initialization of DMA mapping (step 828). Even assuming (for the sake of argument) that initialization of DMA mapping could be properly understood as an indication,

initialization of DMA mapping is certainly not the specific indication recited in claim 19, namely "providing an indication in the virtual memory data structure for the process that a virtual address space is no longer available for use by the process".

4. Conclusion

As explained above, the proposed combination of *Armilli* in view of *Browning et al.* does not teach at least the features noted above and recited in claim 19. Therefore, a *prima facie* case establishing an obviousness rejection has not been made, and the rejection should be withdrawn.

E. Independent Claim 22

In rejecting independent claim 22, the Office Action (p. 11) states that "the limitations of the claim are rejected as the same reasons as set forth in claim 1" even though a quick perusal of the claims shows they are not coextensive in scope. For example, claim 22 recites "registering an indication…in a manner which does not violate semantics of an operating system", and this feature is not present in claim 1. Since the Office Action fails to address all the features recited in claim 22, the rejection is incomplete and deficient. This rejection should be withdrawn and a new rejection issued in another ***non-final Office Action***.

Nonetheless, in the interest of advancing prosecution, Applicant now discusses some distinctions between claim 22 and the proposed combination of *Armilli* in view of *Browning et al.* In doing so, Applicant makes reference to the rejection of claim 1.

1. The proposed combination does not teach "registering an indication in a virtual memory data structure for the process"

The Office Action acknowledges (pp. 3-4) that *Armilli* does not specifically teach this feature, but contends that *Browning et al.* teaches this feature as follows:

> However, Browning teaches a method for invalidating specified pre-translations maintained in a data processing system which maintains decentralized copies of pre-translations, wherein the data processing system comprising a DMA mapping agent to map a virtual buffer to physical address (steps 802-8 16, figure 8), when a detection on memory

> removal operation is being process (col. 7 lines 21 through col. 8 line 44), and also defined in figure 8 that determining whether the RPN entry for the virtual memory page is valid (step 824), when flag is set on memory removal process (step 834), i.e., registering is triggered by detection that the physical address space that was being used by process associated with the device is being released, and the virtual memory page is maintained for mapping as defined in step 828, and further comprising the steps of to register **by providing an indication in the virtual memory data structure for the process that the virtual address space, previously available to the process, is no longer valid for process use, as defined in steps 824-830 in figure 8, and (step 910, figure 9 and col. 8 lines 58-64, i.e., scan all registered RPN lists and invalidates all entries, including virtual address space, corresponding to real pages that within the range of memory to be removed)**, and (col. 8 lines 53-58, i.e., receiving acknowledgement of interrupt from all CPUs and then triggering to register by detecting that real pages that are within range of memory to be removed).
> (Office Action, p.4, emphasis added.)

Applicant respectfully disagrees with this characterization of *Browning et al.*, and submits that none of the passages in *Browning et al.* that are relied on in the rejection disclose a "virtual memory data structure for ***the process***".

It appears that the Office Action relies on the RPN (real page number) data structure as teaching the claimed "virtual memory data structure", and Applicant assumes (for the sake of argument) that the RPN is a virtual data memory structure. Even so, *Browning et al.* does not teach or suggest that the RPN is "for the process" as recited in claim 22. In fact, *Browning et al.* appears to suggest that the RPN is <u>not</u> for a particular process:

> The RPN lists are private copies of virtual to physical translations. The lists are maintained in memory descriptors that are associated and maintained with virtual buffers. Pointers to virtual buffers may be passed from one entity to another. When a pointer to a virtual buffer is passed from one entity to another, control of the RPN list included in the buffer's descriptor is thus passed from one entity to another. In this manner, the lists are not owned by any particular entity. An entity may be a software process, such as a routine, application, or operating system function, or it may be a subsystem.
> (*Browning et al.*, Col. 4, lines 3-13.)

If the rejection is maintained in another Office Action, the Examiner is requested to point out with particularity which data structure in *Browning et al.* allegedly corresponds to "virtual memory data structure for the process".

2. The proposed combination does not teach "registering an indication…that the virtual address space is not available to the process"

The Office Action acknowledges (p. 6) that *Armilli* does not specifically teach this feature, but contends that *Browning et al.* teaches this feature as follows:

> However, Browning teaches a method for invalidating specified pre-translations maintained in a data processing system which maintains decentralized copies of pre-translations, wherein the data processing system comprising a DMA mapping agent to map a virtual buffer to physical address (steps 802-8 16, figure 8), when a detection on memory removal operation is being process (col. 7 lines 21 through col. 8 line 44), and also defined in figure 8 that determining whether the RPN entry for the virtual memory page is valid (step 824), when flag is set on memory removal process (step 834), i.e., registering is triggered by detection that the physical address space that was being used by process associated with the device is being released, and the virtual memory page is maintained for mapping as defined in step 828, and further comprising the steps of to register **by providing an indication in the virtual memory data structure for the process that the virtual address space, previously available to the process, is no longer valid for process use, as defined in steps 824-830 in figure 8, and (step 910, figure 9 and col. 8 lines 58-64, i.e., scan all registered RPN lists and invalidates all entries, including virtual address space, corresponding to real pages that within the range of memory to be removed)**, and (col. 8 lines 53-58, i.e., receiving acknowledgement of interrupt from all CPUs and then triggering to register by detecting that real pages that are within range of memory to be removed).
> (Office Action, pp. 6-7, emphasis added.)

Applicant respectfully disagrees with this characterization of *Browning et al.* It appears that the Office Action relies on the RPN (real page number) data structure as teaching the claimed "virtual memory data structure", and Applicant assumes (for the sake of argument) that the RPN is a virtual data memory structure. Even so, *Browning et al.* does not teach or suggest that the RPN provides an indication that "that the virtual address space is not available to the process" as recited in claim 22.

Browning et al. appears to disclose that an RPN is associated with virtual addresses, through the memory descriptor 50 which includes the RPN (See FIG. 3A.) *Browning et al.* also describes scanning entries in multiple RPN lists and invalidating these entries as part of a kernel memory remove operation. (Col. 8, lines 45-50.) However, Applicant submits that an invalid

RPN entry is not an indication that the virtual address space associated with the RPN is no longer available for use. Instead, an invalid RPN entry merely signifies that the virtual-to-physical mapping contained in the RPN is invalid such that the mapping must be updated. (See FIG. 8, blocks 824 and 832: if RPN entry is invalid, retranslate buffer page.)

> The RPN lists are private copies of virtual to physical translations. The lists are maintained in memory descriptors that are associated and maintained with virtual buffers. Pointers to virtual buffers may be passed from one entity to another. When a pointer to a virtual buffer is passed from one entity to another, control of the RPN list included in the buffer's descriptor is thus passed from one entity to another. In this manner, the lists are not owned by any particular entity. An entity may be a software process, such as a routine, application, or operating system function, or it may be a subsystem.
> (*Browning et al.*, Col. 4, lines 3-13.)

If the rejection is maintained in another Office Action, the Examiner is requested to clearly explain how the RPN in *Browning et al.* allegedly indicates "that the virtual address space is not available to the process".

3. Conclusion

As explained above, the proposed combination of *Armilli* in view of *Browning et al.* does not teach at least the features noted above and recited in claim 22. Therefore, a *prima facie* case establishing an obviousness rejection has not been made, and the rejection should be withdrawn.

F. Independent Claim 23

In rejecting independent claim 23, the Office Action (p. 11) states that "the limitations of the claim are rejected as the same reasons as set forth in claim 19" even though a quick perusal of the claims shows they are not coextensive in scope. For example, claim 23 recites "registering an indication…in a manner which does not violate semantics of an operating system", and this feature is not present in claim 19. Since the Office Action fails to address all the features recited in claim 23, the rejection is incomplete and deficient. This rejection should be withdrawn and a new rejection issued in another ***non-final Office Action***.

Nonetheless, in the interest of advancing prosecution, Applicant now discusses some distinctions between claim 23 and the proposed combination of *Armilli* in view of *Browning et al.* Since the rejection of claim 23 relies on the analysis of claim 19, and that analysis in turn relies on the analysis of claim 8 (see Office Action, p. 11), Applicant makes reference to the rejection of claim 8.

1. The proposed combination does not teach "registering an indication in a virtual memory data structure for the process"

The Office Action acknowledges (p. 6) that *Armilli* does not specifically teach this feature, but contends that *Browning et al.* teaches this feature as follows:

> However, Browning teaches a method for invalidating specified pre-translations maintained in a data processing system which maintains decentralized copies of pre-translations, wherein the data processing system comprising a DMA mapping agent to map a virtual buffer to physical address (steps 802-8 16, figure 8), when a detection on memory removal operation is being process (col. 7 lines 21 through col. 8 line 44), and also defined in figure 8 that determining whether the RPN entry for the virtual memory page is valid (step 824), when flag is set on memory removal process (step 834), i.e., registering is triggered by detection that the physical address space that was being used by process associated with the device is being released, and the virtual memory page is maintained for mapping as defined in step 828, and further comprising the steps of to register **by providing an indication in the virtual memory data structure for the process that the virtual address space, previously available to the process, is no longer valid for process use, as defined in steps 824-830 in figure 8, and (step 910, figure 9 and col. 8 lines 58-64, i.e., scan all registered RPN lists and invalidates all entries, including virtual address space, corresponding to real pages that within the range of memory to be removed)**, and (col. 8 lines 53-58, i.e., receiving acknowledgement of interrupt from all CPUs and then triggering to register by detecting that real pages that are within range of memory to be removed).
> (Office Action, pp. 6-7, emphasis added.)

Applicant respectfully disagrees with this characterization of *Browning et al.*, and submits that none of the passages in *Browning et al.* that are relied on in the rejection disclose a "virtual memory data structure for ***the process***".

It appears that the Office Action relies on the RPN (real page number) data structure as teaching the claimed "virtual memory data structure", and Applicant assumes (for the sake of

argument) that the RPN is a virtual data memory structure. Even so, *Browning et al.* does not teach or suggest that the RPN is "for the process" as recited in claim 23. In fact, *Browning et al.* appears to suggest that the RPN is not for a particular process:

> The RPN lists are private copies of virtual to physical translations. The lists are maintained in memory descriptors that are associated and maintained with virtual buffers. Pointers to virtual buffers may be passed from one entity to another. When a pointer to a virtual buffer is passed from one entity to another, control of the RPN list included in the buffer's descriptor is thus passed from one entity to another. In this manner, the lists are not owned by any particular entity. An entity may be a software process, such as a routine, application, or operating system function, or it may be a subsystem.
> (*Browning et al.*, Col. 4, lines 3-13.)

If the rejection is maintained in another Office Action, the Examiner is requested to point out with particularity which data structure in *Browning et al.* allegedly corresponds to "virtual memory data structure for the process".

2. The proposed combination does not teach "registering an indication…that the virtual address space is not available to the process"

The Office Action acknowledges (p. 6) that *Armilli* does not specifically teach this feature, but contends that *Browning et al.* teaches this feature as follows:

> However, Browning teaches a method for invalidating specified pre-translations maintained in a data processing system which maintains decentralized copies of pre-translations, wherein the data processing system comprising a DMA mapping agent to map a virtual buffer to physical address (steps 802-816, figure 8), when a detection on memory removal operation is being process (col. 7 lines 21 through col. 8 line 44), and also defined in figure 8 that determining whether the RPN entry for the virtual memory page is valid (step 824), when flag is set on memory removal process (step 834), i.e., registering is triggered by detection that the physical address space that was being used by process associated with the device is being released, and the virtual memory page is maintained for mapping as defined in step 828, and further comprising the steps of to register **by providing an indication in the virtual memory data structure for the process that the virtual address space, previously available to the process, is no longer valid for process use, as defined in steps 824-830 in figure 8, and (step 910, figure 9 and col. 8 lines 58-64, i.e., scan all registered RPN lists and invalidates all entries, including virtual address space, corresponding to real pages that within the range of memory to be removed)**, and (col. 8 lines 53-58, i.e., receiving acknowledgement of

> interrupt from all CPUs and then triggering to register by detecting that real pages that are within range of memory to be removed).
> (Office Action, pp. 6-7, emphasis added.)

Applicant respectfully disagrees with this characterization of *Browning et al.* It appears that the Office Action relies on the RPN (real page number) data structure as teaching the claimed "virtual memory data structure", and Applicant assumes (for the sake of argument) that the RPN is a virtual data memory structure. Even so, *Browning et al.* does not teach or suggest that the RPN provides an indication that "that the virtual address space is not available to the process" as recited in claim 23.

Browning et al. appears to disclose that an RPN is associated with virtual addresses, through the memory descriptor 50 which includes the RPN (See FIG. 3A.) *Browning et al.* also describes scanning entries in multiple RPN lists and invalidating these entries as part of a kernel memory remove operation. (Col. 8, lines 45-50.) However, Applicant submits that an invalid RPN entry is not an indication that the virtual address space associated with the RPN is no longer available for use. Instead, an invalid RPN entry merely signifies that the virtual-to-physical mapping contained in the RPN is invalid such that the mapping must be updated. (See FIG. 8, blocks 824 and 832: if RPN entry is invalid, retranslate buffer page.)

> The RPN lists are private copies of virtual to physical translations. The lists are maintained in memory descriptors that are associated and maintained with virtual buffers. Pointers to virtual buffers may be passed from one entity to another. When a pointer to a virtual buffer is passed from one entity to another, control of the RPN list included in the buffer's descriptor is thus passed from one entity to another. In this manner, the lists are not owned by any particular entity. An entity may be a software process, such as a routine, application, or operating system function, or it may be a subsystem.
> (*Browning et al.*, Col. 4, lines 3-13.)

If the rejection is maintained in another Office Action, the Examiner is requested to clearly explain how the RPN in *Browning et al.* allegedly indicates "that the virtual address space is not available to the process".

3. Conclusion

As explained above, the proposed combination of *Armilli* in view of *Browning et al.* does not teach at least the features noted above and recited in claim 23. Therefore, a *prima facie* case establishing an obviousness rejection has not been made, and the rejection should be withdrawn.

G. Dependent Claims 2-7, 9-12, 14-18, and 20-21

Since independent claims 1, 8, 13, and 19 are allowable, Applicant respectfully submits that claims 2-7, 9-12, 14-18, and 20-21 are allowable for at least the reason that each depends from an allowable claim. *In re Fine*, 837 F.2d 1071, 5 U.S.P.Q.2d 1596, 1598 (Fed. Cir. 1988). Therefore, Applicant respectfully requests that the rejection of claims 2-7, 9-12, 14-18, and 20-21 be withdrawn.

CONCLUSION

Applicant respectfully requests that all outstanding objections and rejections be withdrawn and that this application and presently pending claims 1-23 be allowed to issue. Any statements in the Office Action that are not explicitly addressed herein are not intended to be admitted. In addition, any and all findings of inherency are traversed as not having been shown to be necessarily present. Furthermore, any and all findings of well-known art and official notice, or statements interpreted similarly, should not be considered well known since the Office Action does not include specific factual findings predicated on sound technical and scientific reasoning to support such conclusions. If the Examiner has any questions or comments regarding Applicant's response, the Examiner is encouraged to telephone Applicant's undersigned counsel.

Respectfully submitted,

By: /Karen G. Hazzah/

Karen G. Hazzah, Reg. No. 48,472

THOMAS, KAYDEN, HORSTEMEYER & RISLEY, L.L.P.

600 Galleria Parkway, SE
Suite 1500
Atlanta, Georgia 30339-5948
Tel: (770) 933-9500
Fax: (770) 951-0933